Transformando Diseños Orientados a Aspectos en Código Orientado a Aspectos

C. Abdelahad, D. Riesco, L. Baigorria, G. Montejano
Departamento de Informática
Universidad Nacional de San Luis
San Luis, Capital, Argentina
Ejército de los Andes 950 – San Luis – San Luis – Argentina
C.P.: 5700
Tel.: 54-02652-424027 – Int. 251
[cabdelah, driesco, flbaigor, gmonte]@unsl.edu.ar

Resumen

La ingeniería de Software sigue en constante evolución, y con ella se van introduciendo nuevos conceptos los cuales logran obtener una programación de más alto nivel. La Programación Orientada a Objetos (POO) ha sido uno de los avances más importantes en ha tenido la ingeniería de software. Sin embargo existen aspectos que entrecruzan y atraviesan todo el sistema (crosscutting concern) y que no pueden ser totalmente separados con esta técnica. En consecuencia, nace la Programación Orientada a Aspectos (POA), programación basada en la POO, la cual brinda un soporte explícito para tratar estos aspectos.

Actualmente no existe un estándar para la construcción de diseños Orientado a Aspecto (OA) ocasionando que cada ingeniero construya su propio diseño.

Acorde a esto, nuestra investigación se centra en la definición de un diseño OA basado en el metamodelo de la OMG utilizando los mecanismos de extensión que provee UML. Siendo este, un aporte agilizando el proceso de desarrollo de software automatizando la construcción del mismo, a través de la elaboración de una herramienta que genere código OA partiendo de un diseño OA.

A partir de esta herramienta es posible construir transformaciones entre modelos. Una de las transformaciones logradas en esta línea de investigación es la generación de código OA AspectJ partiendo de un diseño OA. Extendiendo el trabajo, se busca lograr construir distintos tipos de

transformaciones desde distintas herramientas de diseño OA basados en

perfiles hacia distintos códigos OA.

Por otro lado se buscó definir un metamodelo en XML con la finalidad de que distintas herramientas, ya sea tanto para el modelado como para la generación de código, puedan interactuar.

Palabras claves: UML, POO, POA, XML, aspecto, perfil, estereotipo.

Contexto

El presente trabajo se enmarca en el Proyecto de Investigación: Ingeniería de Software, Conceptos, Métodos y Herramientas en un Contexto de "Ingeniería de Software en Evolución" – Facultad de Ciencias Físico-Matemáticas y

Naturales, Universidad Nacional de San Luis.

Introducción

El campo de la Ingeniería de Software está en constante evolución. Cada día surgen nuevas técnicas y metodologías que intentan mejorar la calidad y la eficiencia del software.

Considerando que el objetivo de la ingeniería de software es construir un producto de software este trabajo tiene como objetivo automatizar la construcción del mismo.

Un proceso de construcción de software efectivo proporciona normas para el desarrollo eficiente de software de calidad. Este proceso está compuesto, a grandes rasgos, de cuatro etapas – Requisitos, Análisis, Diseño e Implementación. Considerando que la etapa de diseño es una aproximación a la implementación, creando una abstracción de la misma, este trabajo muestra como un diseño detallado está directamente relacionado con los lenguajes de programación y en esta etapa, los modelos se construyen dependiendo del tipo de lenguaje de programación que se utilizará para el desarrollo del software. Esto permite la utilización de tecnologías como la generación de código y la ingeniería de ida y vuelta entre el diseño y la implementación.

Por otro lado, la POO es uno de los avances más importantes en la ingeniería de software para la construcción de sistemas complejos utilizando el principio de descomposición y la reutilización, entre otros. Las descomposiciones poseen el inconveniente de que muchas veces se tienen ejecuciones ineficientes. Éstas surgen debido a que las unidades de descomposición no siempre van acompañadas de un buen tratamiento de los aspectos tales como: sincronización, manejo de errores y manejo de excepciones, administración de memoria y gestión de seguridad entre otros.

La POA [8] es un nuevo tipo de programación, el cual está basado en la POO y brinda un soporte explícito para tratar estos aspectos que entrecruzan y atraviesan todo el sistema (crosscutting concern) [6] y que no pueden ser totalmente separados con las técnicas de tradicionales. AspectJ [11] es un lenguaje creado para el soporte de aspectos. Dado que este lenguaje es una extensión del lenguaje de programación Java, todos los programas válidos en Java también son programas válidos en AspectJ.

UML es un lenguaje que sirve para modelar la mayoría de los dominios, pero no todos los dominios son factibles de ser modelados en este lenguaje.

Para solventar esta desventaja, UML incluye características de extensión. Una de las formas de extender dicho lenguaje es a través de estereotipos. Un estereotipo amplía el vocabulario del UML, permitiendo crear nuevos tipos de bloques de construcción parecidos a los existentes, pero que son específicos a un problema. El uso de definición de estereotipos en la arquitectura de Perfiles del estándar UML [9] permite extender los modelos para diseños orientado a aspectos. De la misma manera que cuando diseñamos un modelo en UML podemos generar código en un lenguaje particular, se puede construir un diseño orientado a aspectos y ser capaces de generar código para distintos tipos de lenguajes orientados a aspectos.

Tanto el diseño como el modelado de sistemas ocupan un lugar importante en la ingeniería de software. Los modelos brindan la posibilidad de abstraer sistemas y facilitar la implementación. La

construcción de buenos modelos asegura un correcto desarrollo de la arquitectura del sistema.

Es imprescindible tener un lenguaje con una sintaxis y semántica precisa para la descripción de un modelo. Existen numerosos lenguajes de modelado. UML es estandarizado por la OMG y es el lenguaje de modelado más conocido y usado en la actualidad. Este lenguaje posee una arquitectura estructurada en capas las cuales son: meta-metamodelo, metamodelo, modelo y objetos. Esta arquitectura a su vez está organizada en paquetes [4].

La ventaja de utilizar un estándar como UML para construir el modelo de diseño es que hay una gran cantidad de herramientas en el mercado que se pueden utilizar. Éstas posibilitan la definición de estereotipos, que podrían ser usados para el soporte de aspectos. Además permiten guardar los documentos como metamodelos en un formato estandarizado como XML [10]. Esto ayuda a que una herramienta que genere código OA sea independiente de las herramientas utilizadas para el modelado.

La importancia de utilizar estándares radica en que garantiza que la herramienta que genere código OA sea portable a distintos tipos de herramientas.

XML permite describir y organizar la información de manera que resulte fácilmente comprensible. XML se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML es una tecnología sencilla que tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

La OMG también da una especificación relacionada con UML y XML la cual permite intercambiar información entre distintas herramientas. Esta especificación lleva el nombre de XMI (XML Metadata Interchange) y está basado en XML.

Líneas de investigación y desarrollo

En la actualidad no existe un estándar para el soporte de aspectos. Por otro lado, la OMG tiene estándares que soportan y permiten la definición de estereotipos a través de los Perfiles. Muchas herramientas soportan este estándar, y a través de estas herramientas es posible definir aspectos utilizando estereotipos para lograr modelos OA. De esta manera es posible la construcción de herramientas que genere código OA totalmente independiente de las herramientas utilizadas para el modelado que sigan con las especificaciones de la OMG.

Un *aspecto* **(aspect)** es la unidad central en un lenguaje OA, de la misma manera que la *clase* es la unidad central en POO.

Por otro lado, los estándares de la OMG brindan la posibilidad de definir Perfiles [2] [3] los cuales hacen posible la construcción de diseños OA, ya que en la actualidad no existe un estándar en cuanto a estos diseños. A través de estos perfiles, es posible definir estereotipos para lograr el soporte de aspectos. La semántica que tendrán estos estereotipos parte de la correspondencia con el código OA, utilizando reglas OCL [1].

Asimismo, muchas herramientas opensource que sigan las especificaciones de la OMG, permiten definir perfiles y de

esta manera se hace posible la construcción de una herramienta que genere código OA totalmente independiente de las herramientas utilizadas para el modelado.

Además, de la misma manera que cuando se diseña un modelo en UML se puede generar código en un lenguaje específico, en este trabajo se muestra cómo es posible construir un diseño *OA* y tener la posibilidad de construir una herramienta la cual tome el diseño OA con formato XMI, generado por la herramienta de modelado, y sea capaz de generar código OA.

Por lo anterior, el interés de nuestra investigación se centró en la definición de un Perfil, basado en el metamodelo de la OMG, orientado a Aspectos [5] para poder construir diseños OA y lograr la posterior construcción de una herramienta, la cual realice la transformación de un diseño OA a código OA. Por otro lado, se aspiró a que la herramienta que genera código OA sea independiente de las herramientas utilizadas para el modelado.

En función de esto, proponemos realizar el estudio de la verificación de lo diseños OA, es decir realizar un chequeo para verificar que el diseño OA está bien construido. Además se aspira a que la herramienta que genera código en AspectJ también permita generar código OA, en el lenguaje COOL para sincronización, o RIDL para distribución, partiendo de un diseño OA.

Resultados y Objetivos

Como mencionamos en el apartado anterior, y siguiendo los objetivos de nuestra investigación, se ha utilizado la definición de un Perfil OA [1] para la construcción de un diseño OA. Partiendo de esto, se construye un metamodelo OA para la posterior elaboración de la herramienta que transforma un diseño OA en código OA, en particular AspectJ.

En la continuidad de este trabajo se prevee profundizar en el estudio de la verificación de lo diseños OA, es decir hacer un chequeo para verificar que el diseño OA está bien construido. Además se aspira a que la herramienta que genere código en AspectJ también pueda permitir generar código OA, en un lenguaje elegido por el usuario tomando como fuente un diseño OA. Por otro lado, se busca que el diseño OA pueda ser construido con cualquier herramienta que trabaje con perfiles, independientemente de la estructura del metamodelo.

Formación de Recursos Humanos

Siguiendo la presente línea de investigación se han finalizado una tesis de posgrado [12] y una tesis de grado [7]. El objetivo de las mismas es proporcionar herramientas que ayuden al ingeniero de software en el desarrollo OA y permitan evaluar el mismo de manera objetiva.

Referencias

[1] N. Debnath L. Baigorria, D. Riesco, G. Montejano “Metrics Applied to Aspect Oriented Design Using UML Profiles” IEEE symposium on computers and communications 2008 (ISCC 2008) Press Marruecos.

[2] OMG Unified Modeling Language specification http://www.omg.org

[3] Jingjun Zhang; Yuejuan Chen; Guangyuan Liu; Modeling Aspect-Oriented Programming with

UMLProfile Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on

[4] UML 2.1.2"Infrastructure Specification" http://www.omg.org/spec/UML/2.2/ 2009

[5] Fuentes L. Vallecillo A., Johson R., Vlissides J. "Una Introducción a los Perfiles UML" Novotica Vol. 168, pg. 6-11. 2004

[6] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J. Loingtier, J. Irwin, "*Aspect-Oriented Programming*". European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. June 1997.

[7] C. Abdelahad "Generación de Código Orientado a Aspectos Partiendo de un Diseño Orientado a Aspectos" Trabajo Final de Licenciatura en Ciencias de la Computación. Universidad Nacional de San Luis, Julio 2010.

[8] Quintero, A. R. Visión General de la Programación Orientada a Aspectos -Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla, 2000.

[9] N.C. Debnath, A. Garis, D. Riesco, G. Montejano "Defining Patterns Using UML Profile"s — Computer Systems and Applications, 2006. IEEE Press.

[10] XML http://www.w3.org/TR/1998/REC-xml-19980210

[11] A. Colyer, A. Clement, G. Harley, M. Webster "Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the Eclipse AspectJ Development Tools". Addison-Wesley Professional.

[12] L. Baigorria "Definición de Métricas en OCL según el Metamodelo de la OMG Aplicadas al Diseño Orientado a Aspectos" Tesis de Maestría en Ingeniería de Software, Universidad Nacional de San Luis, Diciembre 2010.